Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

ORIGINAL

PATENT APPLICATION

ATTORNEY DOCKET NO. 200304471-1

IFW

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s): Carl A. WALDSPURGER et al.

Confirmation No.: 4899

Application No.: 09/401,616

Examiner: M. J. Steelman

Filing Date: 09/22/1999

Group Art Unit: 2122

Title: EFFICIENT, TRANSPARENT, AND FLEXIBLE VALUE SAMPLING

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith in **triplicate** is the Appeal Brief in this application with respect to the Notice of Appeal filed on **12/09/2003**. Also enclosed is an Amendment filed concurrently with Appeal Brief.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $330.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( ) one month	$110.00
( ) two months	$420.00
( ) three months	$950.00
( ) four months	$1480.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of **$330.00**. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: **02/02/2004**

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number_____ on _____

Number of pages:

Typed Name: **Christina L. Paz**

Signature: Christina Paz

Respectfully submitted,

Carl A. WALDSPURGER et al.

By Jonathan M. Harris

Jonathan M. Harris

Attorney/Agent for Applicant(s)
Reg. No. **44,144**

Date: **02/02/2004**

Telephone No.: **(713) 238-8045**

ORIGINAL

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Carl A. WALDSPURGER et al.	Confirmation No.:	4899
Serial No.:	09/401,616	Group Art Unit:	2122
Filed:	09/22/1999	Examiner:	M.J. Steelman
For:	Efficient, Transparent, and Flexible Value Sampling	Docket No.:	200304471-1

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Date: February 2, 2004

Sir:

Appellant hereby submits this Appeal Brief in connection with the above-identified application. A Notice of Appeal was filed on December 9, 2003.

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Company through its merger with Compaq Computer Corporation which owned Compaq Information Technologies Group, L.P. (CITG).

II. RELATED APPEALS AND INTERFERENCES

Appellant is unaware of any related appeals or interferences.

III. STATUS OF THE CLAIMS

Originally filed claims:	1-72.
Claim cancellations:	None.
Added claim:	None.
Presently pending claims:	1-72.
Presently appealed claims:	1-72.

IV. STATUS OF THE AMENDMENTS

Concurrently with this Appeal Brief, Applicants file an Amendment to the specification. The Amendment is irrelevant to the issues in this Appeal.

V. SUMMARY

Hardware and software engineers often desire to collect performance data associated with a computer system. Such performance data can be used to assess how new computer hardware operates with existing operating systems and application programs. For example, compiler writers may like to determine how a compiler schedules instructions for execution, or how well execution of conditional branches are predicted to provide input for software optimization. Similarly, system designers may desire to understand the performance of the operating system kernel, device drivers and application software programs. In general, such performance information helps identify performance problems and facilitates both manual tuning and automated optimization. However, accurately monitoring the performance of hardware and software systems without disturbing the operating environment of the computer system may be difficult, particularly if the performance data is collected over extended periods of time, such as days, or weeks. Applicants' Disclosure, page 1.

In accordance with Applicants' preferred embodiments, the performance of an executing computer program on a computer system is monitored using value sampling. At internals, the execution of the computer program is interrupted, including delivering a first interrupt. In response to at least a subset of the first interrupts, at least one data value of interest is stored in a database. The data value of interest is associated with a particular object code instruction of the program. The object code instruction may be the interrupted instruction. The program itself is not modified by the value sampling process. Applicants' Disclosure, pages 3 and 9.

Applicants have selected claim 1 to discuss with regard to the Examiner's rejections. Claim is as follows:

1. A method for value sampling for monitoring the performance of a program being executed on a computer system, comprising the steps of:

executing the program on a computer system, the program having object code instructions;

at intervals interrupting execution of the program, including delivering a first interrupt; and

in response to at least a subset of the first interrupts, storing at least one data value of interest in a first database, the at least one data value of interest being associated with a particular object code instruction of the program, the particular object code instruction being executed by the computer when the interrupt occurs, wherein the particular object code instruction of the program remains unmodified.

VI. ISSUE(S)

Whether the Examiner erred in rejecting claim 1 over the art of record. For purposes of this Appeal Brief, the patentability of all other claims turns on the patentability issue regarding claim 1.

VII. GROUPING OF CLAIMS

The Examiner rejected the claims as follows:

(1) Claims 1-12, 40, 41, 45, 54, 58-63, 66, 69, and 70 as anticipated by Agrawal (U.S. Pat. No. 5,768,500);

(2) Claims 1-10, 12-14, 22-41, 45-47, 51-54, 66, 68, 69, and 72 as anticipated by Chrysos (U.S. Pat. No. 6,195,748);

(3) Claim 15 as obvious over Chrysos in view of Dean (U.S. Pat. No. 6,237,073);

(4) Claims 16-21, 67 and 71 as obvious over Agrawal in view of Griesemer (U.S. Pat. No. 6,192,516);

(5) Claims 42-44 as obvious by Agrawal in view of Dean;

(6) Claims 48-50 as obvious by Agrawal in view of Griesemer and in further view of Salas (U.S. Patent No. 6,233,600);

(7) Claims 55-57 as obvious by Agrawal in view of Salas; and

(8) Claims 64 and 65 as obvious by Agrawal in view of Gibbons, P.B., et al., "New Sampling-Based Summary Statistics for Improving Approximate Query Answers," proc. ACM SIGMOD, 1998.

Applicants consider claim 1 to be representative of all of the claims for purposes of discussing patentability. As such, for purposes of this appeal brief, Applicants explain why the Examiner erred in rejecting claim 1.

VIII. ARGUMENT

The Examiner erred in rejecting claim 1 as being separately anticipated by Agrawal and by Chyrsos. Both references are summarized below.

A. The Agrawal Reference

Agrawal discloses a memory system profiler that samples system state using interrupts generated by a cache miss counter, compare register, and interrupt line. See Agrawal at col. 2, lines 37-41. In Agrawal, an instruction resulting in a cache miss causes the cache miss counter to be incremented (or decremented). See Agrawal at col. 8, line 34 – col. 9, line 44. The cache miss counter then is checked to see if the counter had reached a predetermined state sufficient to generate an interrupt. If so, an interrupt is generated. *Id.* In the meantime, however, the execution of instructions within the processor proceeds. As a result, the interrupt will occur at a time later than the actual instruction that caused the cache miss and generated the interrupt.

B. The Chrysos Reference

Chrysos discloses a system that allows for sampling of data associated with a particular instruction. Chrysos teaches selecting a particular instruction to be sampled, modifying the instruction to indicate its selection, and then collecting data associated with the instruction as the instruction progresses through the

execution pipeline. See Chrysos at col. 17 lines 29-43. The collected data can then be stored upon completion of the execution of the particular instruction. *Id.* The selected instruction is modified by the addition of a sample field to identify the instruction during execution for sampling. See Chrysos at col. 9, lines 17-20 and col. 13, lines 31-37.

C. The Examiner erred in rejecting claim 1 over Agrawal and over Chrysos

Claim 1 recites in part "storing at least one data value of interest in a first database, the at least one data value of interest being associated with a particular object code instruction of the program, the particular object code instruction being executed by the computer when the interrupt occurs." Agrawal does not teach or suggest the association of the sampled information with the particular object code instruction of the program being executed by the computer when the interrupt occurs. At least for this reason, the Examiner erred in rejecting claim 1 over Agrawal.

Claim 1 also specifies that "the particular object code instruction of the program remains unmodified." As explained above, Chrysos has a completely opposite teaching in that a selected instruction is modified by the addition of a sample field to identify the instruction during execution for sampling. See Chrysos at col. 9, lines 17-20 and col. 13, lines 31-37. At least for this reason, the Examiner erred in rejecting claim 1 over Chrysos.

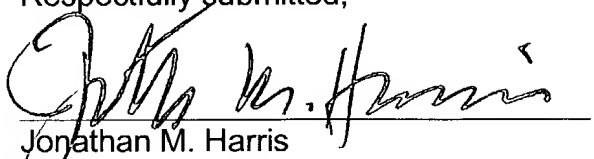
IX. CONCLUSION

For the reasons stated above, the Examiner erred in rejecting claim 1. All claims which depend on or from claim 1 are patentable at least for the same reason as claim 1. Further, the Examiner erred in rejecting all other independent and associated dependent claims at least for the same reasons explained above regarding claim 1.

If any fees or time extensions are inadvertently omitted or if any fees have been overpaid, please appropriately charge or credit those fees to Hewlett-

Packard Company Deposit Account Number 08-2025 and enter any time extension(s) necessary to prevent this case from being abandoned.

Respectfully submitted,



Jonathan M. Harris
PTO Reg. No. 44,144
CONLEY ROSE, P.C.
(713) 238-8000 (Phone)
ATTORNEY FOR APPLICANTS

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
Legal Dept., M/S 35
P.O. Box 272400
Fort Collins, CO 80527-2400

APPENDIX TO APPEAL BRIEF

CURRENT CLAIMS

1. A method for value sampling for monitoring the performance of a program being executed on a computer system, comprising the steps of:

executing the program on a computer system, the program having object code instructions;

at intervals interrupting execution of the program, including delivering a first interrupt; and

in response to at least a subset of the first interrupts, storing at least one data value of interest in a first database, the at least one data value of interest being associated with a particular object code instruction of the program, the particular object code instruction being executed by the computer when the interrupt occurs, wherein the particular object code instruction of the program remains unmodified.

2. The method of claim 1 wherein the intervals are random intervals.

3. The method of claim 1 wherein the intervals have a constant period.

4. The method of claim 1 further comprising the step of:

specifying an object code instruction of interest, and wherein the at least one data value of interest is associated with the specified object code instruction of interest.

5. The method of claim 1 wherein the data value of interest is an operand.

6. The method of claim 1 wherein the data value of interest is a result of the execution of the instruction.

7. The method of claim 1 wherein the particular object code instruction is associated with a memory address, the memory address being stored in a program counter, and wherein said step of storing stores the associated memory address and the at least one data value of interest in the database.

8. The method of claim 1 wherein the program includes object code instructions corresponding to a shared library, and said step of storing stores at least one data value of interest associated with at least one object code instruction derived from the shared library.

9. The method of claim 1 wherein the program includes object code instructions corresponding to at least one kernel instruction, and said step of storing stores at least one data value of interest associated with at least one kernel instruction.

10. The method of claim 1 further comprising the step of:
specifying a set of object code instructions of interest, and
wherein said step of storing is performed only when the program is interrupted at a point associated with at least one of the set of object code instructions of interest.

11. The method of claim 1 wherein said step of storing stores a current interrupt level.

12. The method of claim 1 wherein said step of storing stores physical addresses for load and store instructions to provide information about the number and sources of memory references .

13. The method of claim 1 wherein said step of storing stores at least one data value that identifies the destination of at least one instruction that transfers control

flow as the at least one data value of interest.

14. The method of claim 13 wherein said at least one instruction that transfers control flow includes any of a conditional branch instruction, an unconditional branch instruction, a jump instruction, a call subroutine instruction, or a subroutine return instruction, and the destination is identified by an associated destination address.

15. The method of claim 13 wherein the at least one instruction is a conditional branch instruction, and the destination is identified by a bit.

16. The method of claim 1 wherein said step of storing includes the steps of:
identifying at least one issue block of instructions;
interpreting the instructions of the at least one issue block; and
storing at least one data value of interest associated with at least one interpreted instruction.

17. The method of claim 16 wherein said step of storing stores the at least one data value of interest after interpreting each instruction of said at least one issue block.

18. The method of claim 16 wherein said step of interpreting emulates a machine language instruction set of the computer system.

19. The method of claim 16 wherein the interpreter updates a state of the interrupted program as though each interpreted instruction had been directly executed by the computer system.

20. The method of claim 16 wherein the interpreter interprets both kernel and user code.

21. The method of claim 16 wherein the interpreter does not execute particular instructions.

22. The method of claim 1 wherein the particular object code instruction is an interrupted instruction, the particular object code instruction being associated with a memory address, and said step of storing includes the steps of:

storing the memory address and the interrupted instruction;

configuring a second interrupt to be delivered after a predetermined number of instructions are executed; and

in response to the second interrupt,

deactivating the second interrupt, and

storing the memory address and the at least one data value of interest for the associated instruction in the first database.

23. The method of claim 22 wherein the predetermined number of instructions is equal to a number of instructions of an issue block.

24. The method of claim 1 wherein the particular object code instruction is an interrupted instruction, the particular object code instruction being associated with a memory address, and said step of storing includes the steps of:

storing the memory address and a set of object code instructions including the interrupted instruction;

configuring a second interrupt to be delivered after a predetermined number of instructions; and

in response to the second interrupt,

deactivating the second interrupt, and

storing the memory address and the at least one data value of interest for the interrupted instruction in the first database.

25. The method of claim 24 further comprising the step of:

analyzing the interrupted instruction to determine the at least one data value of interest to store in the first database.

26. The method of claim 1 wherein the particular object code instruction is associated with a set of object code instructions, the set of object code instructions also including an interrupted instruction, the interrupted instruction also being associated with a memory address, and said step of storing includes the steps of:

storing the memory address and the set of object code instructions including the interrupted instruction;

configuring a second interrupt to be delivered after a predetermined number of events; and

in response to the second interrupt,

deactivating the second interrupt, and

storing the memory address and the at least one data value of interest for at least one instruction of the set of object code instructions in the first database.

27. The method of claim 26 wherein the set of object code instructions is an issue block.

28. The method of claim 26 wherein the predetermined number of events is a predetermined number of instruction executions.

29. The method of claim 26 wherein the predetermined number of events is a predetermined number of clock cycles.

30. The method of claim 1 further comprising the steps of:

periodically storing the at least one data value of interest of the first database in a second database.

31. The method of claim 30 further comprising the step of:

generating at least one value profile for the at least one data value of interest in the second database.

32. The method of claim 1 wherein the at least one data value of interest has a plurality of data values of interest, and said step of storing stores a tuple of the plurality of the data values of interest and a memory address in the first database.

33. The method of claim 32 wherein one of the data values of interest is a particular data value of interest and the tuple includes a count that is associated with the particular data value of interest, and

said step of storing further includes the steps of:

identifying the particular data value of interest in the first database; and

incrementing the count associated with the particular data value of interest.

34. The method of claim 32 wherein the data values of interest stored in the tuple include a value stored in a register specified by the interrupted instruction.

35. The method of claim 32 wherein the data values of interest stored in the tuple include the value stored in a destination register specified by the instruction and at least one other value stored in another register.

36. The method of claim 1 wherein the particular object code instruction is part of an issue block of instructions, and said step of storing stores one or more data values of interest for one or more instructions of the issue block, the data values of interest including a value of a register specified by one of the instructions of the issue block, including a value stored in a return address register and a value stored in a memory location in a current stack frame.

37. The method of claim 32, wherein one of the data values of interest is a particular data value of interest and the tuple is a particular tuple having particular values of interest, the tuple also having a count that is associated with all the particular data values of interest of the particular tuple, and said step of storing further includes the steps of:

identifying the particular tuple with the particular data values of interest in the first database based on the at least one data value of interest; and

incrementing the count associated with the particular tuple when the at least one data value of interest is the same as the particular data values of interest of the particular tuple.

38. The method of claim 1 wherein said step of storing stores a program counter value which invoked a function containing the profiled instruction and a value stored in a destination register as correlated values.

39. The method of claim 1 wherein said step of storing stores a return address which invoked a function containing the profiled instruction and a value stored in a destination register as correlated values, and further comprising the step of:

determining if the correlated values are already stored in the first database, and if so incrementing a counter associated with the correlated values, otherwise storing the correlated values in the first database, whereby the correlated values represent a profile of information to call sites.

40. The method of 1 further comprising the step of:

optimizing the program of object code instructions based on the at least one data value of interest.

41. The method of claim 1 wherein the at least one data value of interest is context information, and said step of optimizing optimizes the program of object code instructions based on the context information.

42. The method of claim 1 wherein said step of storing stores a set of particular data values of interest for the particular object code instruction, and further comprising the steps of:

generating another program of object code instructions from a set of object code instructions such that separate, specialized versions of object code are generated when at least one of the set of particular data values of interest has a predetermined amount of invariance.

43. The method of claim 42 wherein the set of particular data values of interest include referenced addresses to a memory, and said step of generating another program of object code instructions generates object code instructions to prefetch instructions from at least one of the referenced addresses when the at least one of the referenced addresses has a sufficient invariance to decrease a latency of a memory reference.

44. The method of claim 42 wherein the set of particular data values of interest include an expected value of an input value for a particular instruction, and said step of generating generates another program of object code instructions to execute the particular instruction using the expected value, and that generates object code instructions that determine an actual value of the input value for the particular instruction, and compares the actual value to the expected value to determine whether the particular instruction was executed with an appropriate value.

45. The method of claim 1 wherein said step of storing includes the step of:

applying a predetermined function to the at least one data value of interest before the data value of interest is stored in the first database.

46. The method of claim 1 further comprising the step of:

identifying an instruction of interest having at least one operand; and said step of storing including the steps of:

generating at least one projected value from the at least one operand by applying a function to the at least one operand of the instruction of interest; and

storing the at least one projected value in the first database.

47. The method of claim 32, wherein one of the data values of interest is a particular data value of interest and the tuple is a particular tuple having particular values of interest, the tuple also storing a functional value, wherein the first database stores a set of tuples that are associated with different particular data values of interest associated with the particular object code instruction,

for the particular object code instruction, updating the functional value in the particular tuple based on the at least one data value of interest associated with the particular object code instruction and at least one data value of interest of each tuple in the set of tuples.

48. The method of claim 16 further including the steps of:

receiving a downloaded script; and

wherein said step of storing includes the steps of:

after interpreting one of the instructions, executing the downloaded script to determine and store the at least one data value of interest.

49. The method of 16 further including the steps of:

receiving an interpretable program having interpretable instructions;

receiving a downloaded script that causes a user-mode trap to be sent to the interpretable program;

executing the interpretable instructions with a virtual machine; and

wherein said step of interpreting interprets said executing interpretable instructions, wherein said interrupted instruction is derived from said interpretable instructions, and

said step of storing includes the step of executing the downloaded script to cause a user-mode trap to be sent to said interpretable program whereby virtual machine specific context information may be stored.

50. The method of 16 further including the steps of:

receiving a program having bytecode instructions;

receiving a downloaded script that causes a user-mode trap to be sent to the program;

executing the bytecode instructions with a virtual machine; and

wherein said step of interpreting interprets said executing bytecode instructions, wherein said interrupted instruction is derived from said bytecode instructions, and

said step of storing includes the step of executing the downloaded script to cause a user-mode trap to be sent to said program whereby virtual machine specific context information may be stored.

51. The method of claim 1 wherein said step of storing includes directly delivering the at least one data value of interest to the interrupted program.

52. The method of claim 51 wherein said step of storing includes generating a user-mode interrupt to deliver the at least one data value of interest to the interrupted program.

53. The method of claim 51 wherein a portion of the object code instructions are executed in a kernel, and said step of storing includes performing an upcall from the kernel to a user-mode handler in the interrupted program to store the at least one data value of interest.

54. The method of claim 1 wherein said step of storing includes the steps of:

identifying a process identifier associated with the program; and

storing the at least one data value of interest in a memory space associated with the process identifier such that only the program can access the at least one data value.

55. The method of claim 1 wherein said step of storing includes the steps of:

identifying access control identifiers associated with the program, particular ones of the access control identifiers also being associated a particular user; and

storing the at least one data value such that only the particular user associated with the access control identifier can access the at least one data value.

56. The method of claim 55 wherein the access control identifier includes at least one of a process identifier, a user identifier and a group identifier.

57. The method of claim 1 wherein said step of storing includes the steps of:

identifying a group identifier associated with the program; and

storing the at least one data value of interest in a user space associated with the group identifier such that only a user associated with the group identifier can access the at least one data value.

58. The method of claim 1 further comprising the steps of:

storing the at least one data value of interest of the first database in a hotlist of most frequently occurring data values, the hotlist storing a predetermined number of data values and a count associated with each data value.

59. The method of claim 58 further comprising the step of encoding the hotlist.

60. The method of claim 59 wherein said step of encoding encodes the hotlist by sorting at least a subset of the data values of the hotlist by the count.

61. The method of claim 59 wherein said step of encoding encodes the hotlist by reordering at least a subset of the data values of the hotlist such that the data values in the at least one subset are stored in contiguous memory locations.

62. The method of claim 58 wherein said step of storing the at least one data value of interest of the first database stores the at least one data value of interest using a randomized technique.

63. The method of claim of claim 62 wherein said step of storing the at least one data value of interest dynamically adapts between a first randomized technique and a second randomized technique.

64. The method of claim 58 wherein said step of storing the at least one data value of interest of the first database in a hotlist of most frequently occurring data values uses a concise samples technique.

65. The method of claim 58 wherein said step of storing the at least one data value of interest of the first database in a hotlist of most frequently occurring data values uses a counting samples technique.

66. A computer system for value sampling a computer program having object code instructions, comprising:

a processor for executing the object code instructions of the computer program; and

a memory for storing instructions that:

deliver interrupts at intervals during execution of the program, including delivering a first interrupt; and

store at least one data value of interest in a first database in response to at least a subset of the first interrupts, the at least one data value of interest being associated with a particular object code instruction of the program, the particular object code instruction being executed by the computer when the interrupt occurs, and wherein the particular object code instruction of the program remains unmodified.

67. The computer system of claim 66 wherein said instructions that store further include instructions that:

identify at least one issue block of instructions;

interpret the instructions of the at least one issue block; and

store at least one data value of interest associated with at least one interpreted instruction.

68. The computer system of claim 66, wherein said instructions that deliver interrupts interrupt a particular object code instruction as an interrupted instruction, the particular object code instruction being associated with a memory address; and

wherein said instructions that store further include instructions that:

store the memory address and the interrupted instruction;

configure a second interrupt to be delivered after a predetermined number of instructions; and

in response to the second interrupt,

deactivate the second interrupt, and

store the memory address and the at least one data value of interest for the associated instruction in the first database.

69. A computer program product for value sampling a computer program having object code instructions, the computer program product for use in conjunction with a computer system, the computer program product comprising a computer readable storage medium and a computer program mechanism embedded therein, the computer program mechanism comprising:

instructions that deliver interrupts at intervals during execution of the program, including delivering a first interrupt;

instructions that, in response to at least a subset of the first interrupts, store at least one data value of interest in a first database, the at least one data value of interest being associated with a particular object code instruction of the program, the particular object code instruction being executed by the computer when the interrupt occurs, and wherein the particular object code instruction of the program remains unmodified.

70. The computer program product of claim 69 wherein the intervals are random intervals.

71. The computer program product of claim 69 wherein the instructions that store further include instructions that:

identify at least one issue block of instructions;

interpret the instructions of the at least one issue block; and

store at least one data value of interest associated with at least one interpreted instruction.

72. The computer program product of claim 69 wherein the instructions that deliver interrupts interrupt a particular object code instruction as an interrupted instruction, the particular object code instruction being associated with a memory address; and

wherein the instructions that store further include instructions that:

store the memory address and the interrupted instruction;

configure a second interrupt to be delivered after a predetermined number of instructions; and

in response tai the second interrupt,

deactivate the second interrupt, and

store the memory address and the at least one data value of interest for the associated instruction in the first database.